SpaCEM^3
User Manual

MISTIS Project (INRIA Grenoble Rhône-Alpes, France) and
SaAB Team (BIA Unit, INRA Toulouse, France)
Contact for help: `SpaCEM3-help@lists.gforge.inria.fr`

November 2010

1 Introduction

SpaCEM^3 (Spatial Clustering with EM and Markov Models) is a software that provides a wide range of supervised or unsupervised clustering algorithms. The main originality of the proposed algorithms is that clustered objects do not need to be assumed independent and can be associated with very high-dimensional measurements. Typical examples include image segmentation where the objects are the pixels on a regular grid and depend on neighbouring pixels on this grid. More generally, the software provides algorithms to cluster multimodal data with an underlying dependence structure accounting for some spatial localisation or some kind of interaction that can be encoded in a graph.

This software, developed by members of Team MISTIS (INRIA Grenoble Rhône-Alpes, France), is the result of several research developments on the subject (see references at the end of this document). The present guide corresponds to version 2.09 of the software which is CeCILLB licenced.

This document is intended to present the software, its main functionalities and an introductory use of the Graphic User Interface (GUI). Some theoretical considerations are developed in Section 6.

2 Main features

The approach is based on the EM algorithm for clustering and on Markov Random Fields (MRF) to account for dependencies. In addition to standard clustering tools based on independent gaussian mixture models, SpaCEM^3 features include:

- The unsupervised clustering of dependent objects. Their dependencies is encoded via a graph not necessarily regular and data sets are modelled via Markov random fields and mixture models (eg. MRF and Hidden MRF). Available Markov models include extensions of the Potts model with the possibility to define more general interaction models.
- The supervised clustering of dependent objects when standard Hidden MRF (HMRF) assumptions do not hold (ie. in the case of non correlated and non unimodal noise models). The learning and test steps are based on recently introduced Triplet Markov models.
- Selection model criteria (BIC, ICL and their mean-field approximations) that select the "best" HMRF according to the data.

- The possibility of producing simulated data from:
  - general pairwise MRF with singleton and pair potentials (typically Potts models and extensions),
  - standard HMRF, ie. with independent noise model,
  - general Triplet Markov models with interaction up to order 2
- A specific setting to account for high-dimensional observations.
- An integrated framework to deal with missing observations, under Missing At Random (MAR) hypothesis, with prior imputation (KNN, mean, ...), online imputation (as a step in the algorithm) or without imputation.

3 Installation:

The software can be downloaded from `http://spacem3.gforge.inria.fr`. A Graphic User Interface (GUI) is available whatever the plateform. After filling the online form (not compulsory, only for information purpose), a window offers to download the desired version depending on the user's system. The installation is then straightforward (32 and 64 bits versions are provided; if you are unsure try the 32 bits one):

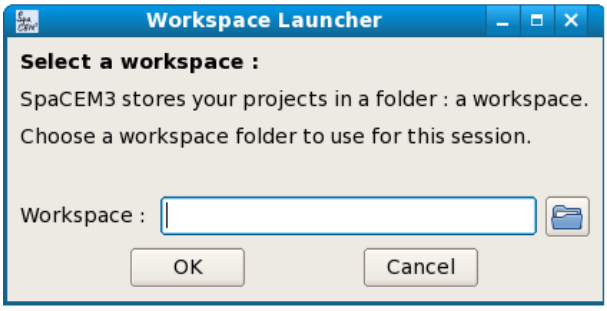
- Windows: double click on the 'Setup' file and then follow the instructions on your screen (administrator rights might be required). Dependent packages are included.
- Linux: with administrator rights ('su+password' or 'sudo+command line'), for common distributions:
  - for Debian or Ubuntu, install the `.deb` package (e.g. `dpkg -i spacem3_version.deb`). Dependencies are included in the package.
  - for Fedora-based distributions, install the `.rpm` package (e.g. `yum install spacem3_version.rpm`). Dependencies (e.g. `apt-get -f install needed-package`) need to be installed indepently. They consists in *qt4*, *graphviz*, *lapack*, *gsl*, *qwt* and *gts* (possibly you'll need to add a *-devel*-like extension; denomination might slightly change from one version to another)
  - for other distribution, a self-extracting archive `.sh` is available. Note that just like for `.rpm` package, dependencies need to be installed manually. Files and folders are installed in the folder where the extraction is run.

  To start the software, run the `guiSpaCEM3` executable file in the `/usr/bin/` folder (or where you decided to place it). This documentation along with example datasets can be found in `/usr/share/doc/spacem3/`.
- MacOS: install the `.pkg` package [not available yet but should be released soon]

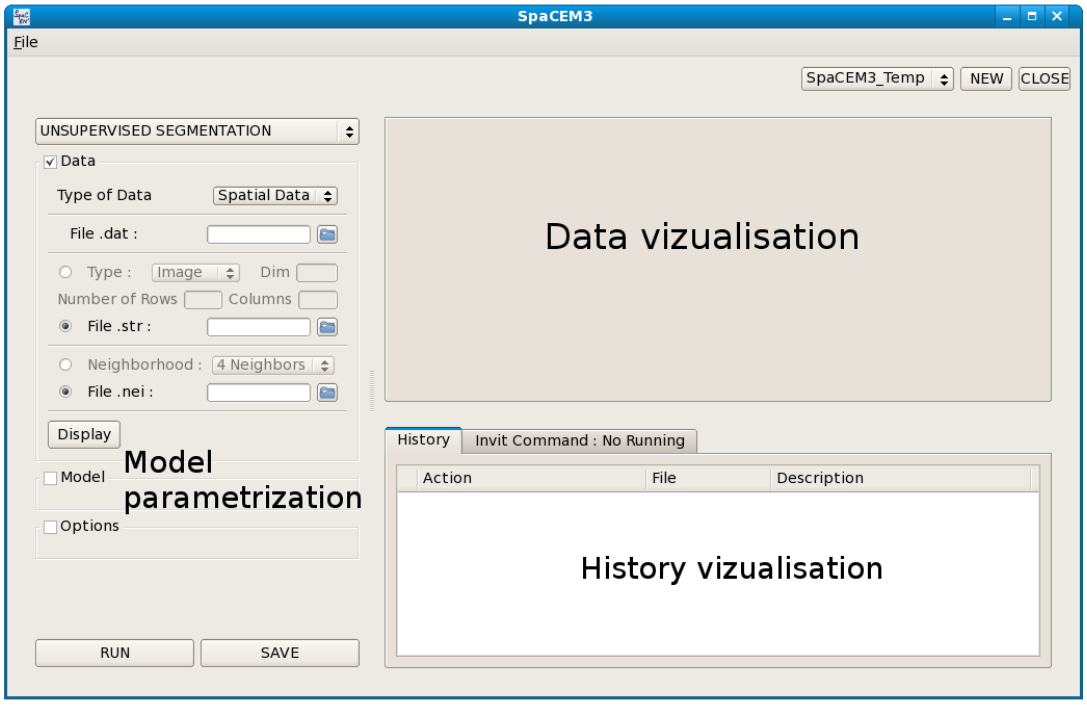
4 Using the GUI of SpaCEM^3

4.1 General principles

When launching the software, a dialog box proposes to the user to specify a working directory. It will contain all output files, in particular results files.


Workspace Launcher
Select a workspace :
SpaCEM3 stores your projects in a folder : a workspace.
Choose a workspace folder to use for this session.
Workspace :
OK
Cancel

Once the workspace has been set, the main SpaCEM^3 interface window is available.



The GUI window is composed of:

- A central part for useful visualization like dataset inspection either with buttons or spinbox. It can for example allow the user to inspect the different profiles, as in Figure 1, by varying the x-axis and the y-axis of the observed object (e.g. pixel or gene), to zoom in and out images and to save the visualization (different formats available: jpeg, png...). With a right-click on the data visualization part, a menu appears with a choice between "Compare: Put Left" and "Compare: Put Right". When a data set is first displayed, it appears at the left-hand side of the data visualization part. The "Compare: Put Right" action allows then to move it to the right. At the first instance of "Compare: Put Right", another tab is opened and this allows the user to compare results by placing them next to each other. The "Compare: Put Left" action as a similar effect but from right to left.

- A left part for data parametrization. A click on the checkbox opens or shuts command groups in the menu.
- A right bottom part composed of two tabs. The "History" tab allows the user to trace actions that were executed. It interacts with the visualization frame so as to display the tab which corresponds to an object that is selected from the list. The "Invit Command" tab tracks the status of a running computation until it is finished. It also provides additional information when an error occurs. On the right of these two tabs, a computation progress bar is visible.

To start a working session, the user has to choose which fonctionality of the software is going to be used:

- Unsupervised clustering (segmentation),
- Supervised clustering (segmentation) or
- Data simulation.

For each of these choices, the corresponding steps will be described in details in the following sections. The general principle is that the data under study has to be entered before designing the model used for the analysis, choosing the estimation approach and then setting relevant options. Once this is achieved, this choice is saved in an .xml file. The core software does the computation and results are stored as output in several files including another .xml file that encodes the structure within the user interface for data visualization.

To start a new work session (e.g. to infer a model on data that has just been simulated as described in Section 4.4), click on the "NEW" button (upper right). A new working space is then created. Navigating between working spaces is made easy with the scrolling button. When a session is not needed anymore, it can be closed ("CLOSE" button next to the "NEW" one).

Several actions are possible *via* the "File" menu (upper left corner):

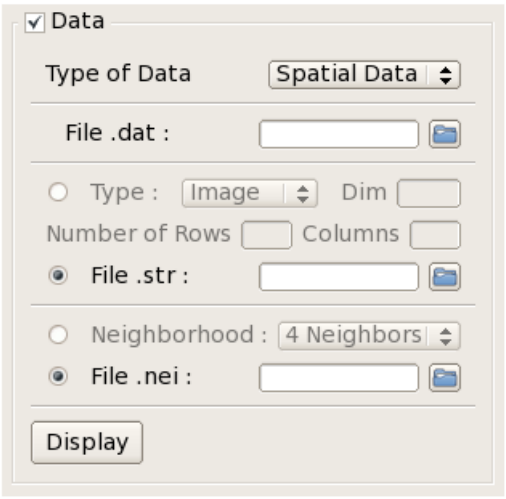
- "Load Model" gives the possibility to upload an existing work with a preset model and tasks to be accomplished; it makes it quicker to run the same computation on different data sets for example.
- "Load Result" allows the user to visualize results obtained in a previous session, for example to compare them with those of the day.

In both cases, the corresponding XML file has to be loaded.

4.2 Unsupervised segmentation

4.2.1 Choosing the data ("Data" group)

Type of data. Dependent variables can be described via a graph where the nodes correspond to the variables and the edges indicate dependent variables. In the current version the software treats separately the case of 3D data (*e.g.* 3D images on a regular 3D grid). For 2D images (on 2D regular grids) and other cases (arbitrary graphs), the type of data has to be set to "Spatial Data". The software accepts two different data formats as an input: ascii files (.dat or .txt file extension) and binary files (`.bin` file extension). The data file has to be given in the "File .dat" blank space.



Ascii files should have the following structure: each row represents data for one sample while each columns give the data value for a given dimension. For example, with N samples in d dimensions,

```
x11 x12 x13 ... x1d
x21 ... x23 ... x1d
....
xN1 xN2 xN3 ... xNd
```

where `xi1 xi2 ... xid` are the d observations for individual i.

When the amount of observations is important, it might be preferable to import them in a binary format to save some time when reading the file. With Matlab, creating a binary file can be done with the following commands:

```
fid=open ('file_name.bin','w')
fwrite(fid,t(X),'float32')
```

where `X` is the vector with observations on a single row:
`x11 x12 x13 ... x1d x21 ... .... xn1 xn2 xn3 ... xnd`

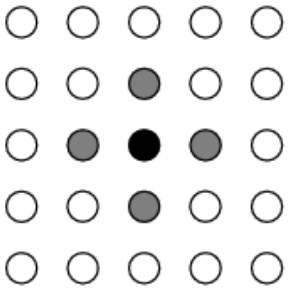
The kind of data at hand needs to be specified: "Image" or "Structure". "Image" formatted data sets are those that are defined on a regular grid like pixels of an image. The numbers of rows and columns have then to be specified. "Structure" formatted data sets are the most general cases of dependent points. The graph that states relationships between individuals is totally customizable. The structure of the data can be specified either manually *via* the graphical interface or in a file (`.str` extension) that gathers the necessary information. More specifically, this file should be composed of the only line:

- `I row_nb column_nb dimension` for the "Image" type. 'I' stands for image, '`row_nb`' indicates the number of lines in the image while '`column_nb`' indicates the number of columns. '`dimension`' is the dimension of the data.
- `S indiv_nb dimension` for the "Structure" type. "S" stands for spatial, '`indiv_nb`' is the sample size (or number of considered objects/individuals), '`dimension`' still being the dimension of the data (*e.g.* the number of time points in a time series).

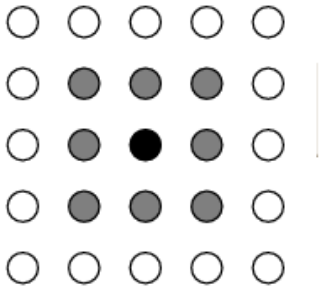
Then, for spatial or dependent data, the user must define the neighbourhood relationships that have to be considered, either manually or within a file (with .nei extension):

- For the "Image" type, regular neighbourhood systems can be set to: (i) the order 1 neighbourhood where each pixel has 4 neighbours, the 4 closest on the grid (North, South, East and West), or (ii) the order 2 neighbourhood where the 8 closest neighbouring pixels (N, S, E, W, NW, NE, SW and SE) are considered.

4 neighbour-grid



8 neighbour-grid



For a 4 neighbour-grid, the .nei file reads:

```
-1 1
-1 1
0 1 0
1 0 1
0 1 0
```

For a 8 neighbour-grid, the `.nei` file reads:

```
-1 1
-1 1
1 1 1
1 0 1
1 1 1
```

- For the "Structure" type, irregular neighbourhood graph can be used. Relationships between objects/individuals need then to be fully specified. This has to be done in a file (`.nei` extension) with the following structure:

```
0
1 nei_nb_1 n11 n12 ...
2 nei_nb_2 n21 n22 ...
3 nei_nb_3 n31 n32 ...

...
```

'0' on the first line stands for unweighted graphs (putting weights on edges has not been implemented yet in this version of the software); indices '1', '2', '3' on the second, third and fourth line respectively correspond to the objects indices; 'nei_nb_i' ($\texttt{i} \in \{1, 2 \ldots, N\}$) defines the number of neighbours of object i in the graph. Then, on each line follows the list of these neighbours: nij is the jth neighbour ($\texttt{j} \in \{1 \ldots \texttt{nei\_nb\_i}\}$) of object i ($\texttt{i} \in \{1, 2, \ldots N\}$).

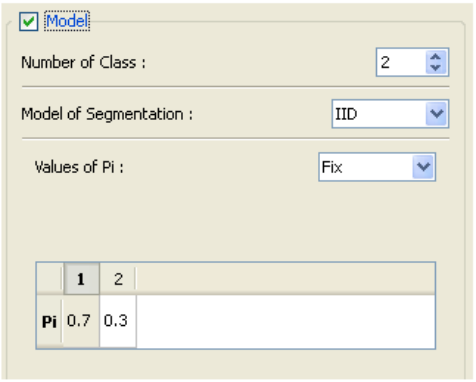
3D (image) data. In the case of 3D data, the user has to specify this setting in the "Type of Data" tab and specify the number of neighbours (on regular 3D grids, the first and second order neighbourhood represents respectively 6 and 26 neighbours). Then, if the 3D data of interest is only partially included in the 3D regular grid, the user needs to specify the 3D locations (X, Y, Z coordinates) in an additional file. This can be useful, for instance, when considering magnetic resonance images of the brain. Data sets are 3D volumes but the data of interest (brain tissues) has to be extracted from the background.

The user can now visualize data thanks to the "Display" button and get some simple summary statistics for them.

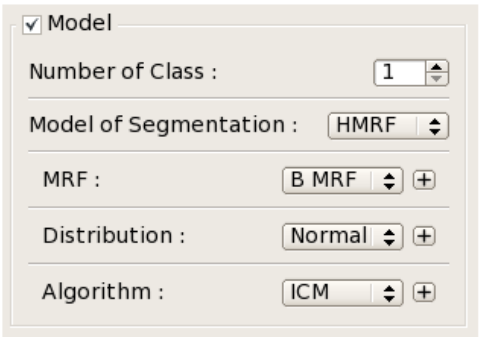
4.2.2 Choosing the model ("Model" group)

Two kinds of models are available in SpaCEM^3: "IID" models where individuals are assumed to be independent and "HMRF" models based on Hidden Markov Random Fields (HMRF) (see Section 6.1 for more details). For both types of models, the first parameter to be specified is the desired number of clusters in the segmentation.

For "IID" models, important parameters are the *a priori* proportions of individuals in each class ("Values of Pi" in the interface). The user can either choose to have these proportions estimated ('Unfix' option), to fix them to the same value with the "Fix Equal" option or to set them one by one with the "Fix" option. The picture to the right shows a case of 2 classes with respective a priori proportions fixed to 0.7 and 0.3.



For "HMRF" models, the kind of Markov Random Field to be considered has to be chosen via the "MRF" tab (see Section 6.1.2 for the available possibilities). The different parameters can also be potentially fixed by the user via the "+" button on the right-hand side when choosing the 'Fix' option. A double-click on the capture area opens a new window that makes it easy to type these values as a vector or as a matrix.



Then, in both the independent and Markovian settings, a distribution that models the class dependent distributions has to be specified. Again, the user must specify whether the corresponding

parameters are estimated or fixed to some prescribed values. For example, in the case of K classes, when choosing Gaussian distributions for the class-dependent distributions ("Normal" option), the user is asked to specify the mean ('Mu') and variance ('Sigma') for the K classes via a $2 \times K$ table. If in addition the data are D-dimensional (*i.e.* the Gaussian distributions must be multivariate), a double click in each cell of the table opens a window that simplifies parameter capture as vectors (*e.g.* D-dimensional vectors for the means) or as matrices (*e.g* $D \times D$ matrices for the variances).

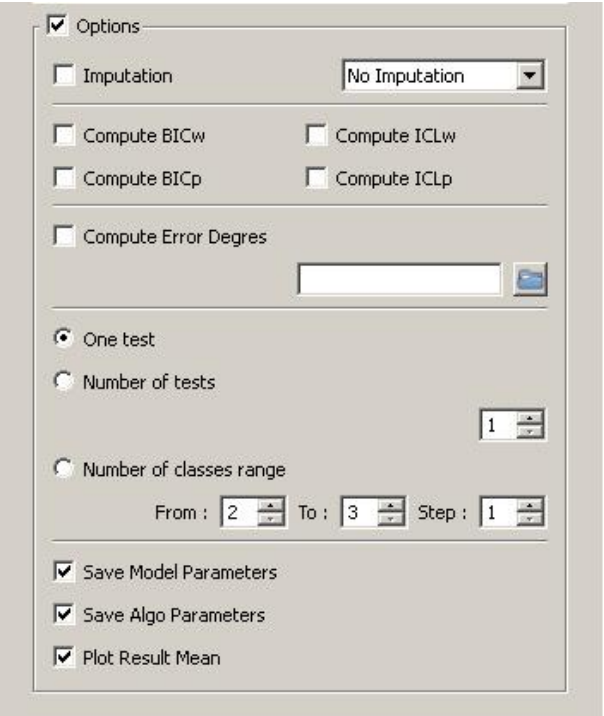
Eventually, the algorithm to estimate the model has to be chosen (see Section 6.2). The initialization procedure can be set by clicking on the "+" symbol at the right-hand side of the "Algorithm" field. Available initializations are "Random" (default) initialization, "KMeans" initialization or via a preset clustering "Fix Init" from a text file. Although "Random" is the default initialisation, we suggest the user use "KMeans" or "Fix Init" for better clustering results. The algorithm stopping criterion can be chosen from a choice of four:

- "Number of iteration" to be performed by the algorithm (100 by default),
- "Completed Likelihood" that stops the algorithm as soon as the difference between two values of the complete likelihood at two successive iterations does not exceed a given threshold,
- "Fuzzy classification" that stops the algorithm when the maximum difference in absolute value between two successive assignment probabilities (*i.e.* probabilities that an object -*e.g.* gene or pixel- is assigned to a given cluster) does not exceed a given threshold, or
- "Hard Classification" that stops the algorithm when the proportion of objects, whose class assignment varies between two successive iterations, is under a given threshold.

In all cases, the 'threshold' can be fixed via the "Tolerance" tab (defaults for the above 4 cases are respectively 100, 100, 0.01, 0.01).

4.2.3 Choosing options

A last optional step before running the computation is about choosing available options.



Available options are the following:

- "Imputation" offers different approaches to handle missing data if some of the observations are missing (see Section 6.4).
- "Compute BIC" computes the BIC criterion ("IID" case) or an approximation ("HMRF" case) as an output of the program. Available approximations are denoted by BIC_w and BIC_p (see Section 6.3).
- "Compute ICL" computes the ICL criterion ("IID" case) or an approximation("HMRF" case) as an output of the program. Available approximations are denoted by ICL_w and ICL_p (see Section 6.3 too).
- "Compute Error Degrees" computes the error rate in classification (1 minus accuracy); a Gold Standard classification file has to be given as an input of the software.
- "Number of tests" launches the same computation several times. Once the different runs are finished, parameter means are displayed (only available in the unsupervised clustering case).
- "Number of classes range" allows to do the same kind of analysis for different numbers of clusters in the given range. Once computations are finished, graphs with different criteria versus the number of clusters are drawn. This can help choosing the "best" model (only available in the unsupervised clustering case).

- "Save Model Parameters" saves the class-dependent distribution parameters and the final fuzzy classification (assignments probabilities) and displays their change over iterations of the algorithm.
- "Save Algo Parameters" saves each stopping criterion values and displays their change over iterations of the algorithm.
- "Plot Result Mean" plots, for each class in the final segmentation, the mean value of the objects assigned to this class. This is especially useful when clustering multidimensional data such as time series or spectra (*e.g.* in remote sensing applications). The plot shows the series or spectra corresponding to the mean in each class. Figure 1 shows an illustration on hyperspectral data from Mars.

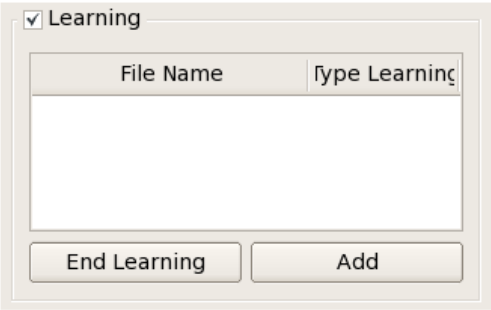
4.3 Supervised segmentation

In a supervised segmentation context, we consider two sets of observed variables respectively indexed by $\mathcal{I}^1$ and $\mathcal{I}^2$. The observed data set denoted by $\mathbf{x}^1 = (x_i)_{i\in\mathcal{I}^1}$ in $\mathcal{I}^1$ corresponds to labelled data for which the assignments are known and denoted by $\mathbf{z}^1 = (z_i)_{i\in\mathcal{I}^1}$. For the other observed data set $\mathbf{x}^2 = (x_i)_{i\in\mathcal{I}^2}$ in $\mathcal{I}^2$ the labels are not known. The goal is then to learn from data $\mathbf{x}^1$ and $\mathbf{z}^1$ (the learning data set) some model parameters so as to be able to assign in a following step the unlabelled data $\mathbf{x}^2$ (test data set). The learning and test data sets are assumed to come from the same parametric model so that for both, we will respectively denote by $\mathbf{X}$ and $\mathbf{Z}$ the observations and corresponding classes.

Supervised segmentation consists of two steps, an initial learning step and a subsequent classification or test step. These steps are briefly described below. To fully understand the modelling capabilities of these options we refer the reader to Section 6.1.4.

4.3.1 Learning step

In the software, the learning step can be carried out as follows. It is assumed that each class k is learned separately, which means that data labelled as belonging to class k must be all gathered in a single file, say 'learn-k.dat'. The previous "IID" and "HMRF" models (Section 4.2.2) are available to perform this learning step which can be carried out using the same steps as described before (see Section 6.2.3 for details).



For each class, the corresponding learning data file as to be entered (*e.g.* 'learn-k.dat') together with a model and an unsupervised segmentation algorithm (Section 6.2). Starting from class 1 and data in file 'learn-1.dat', choose the model and algorithm to be performed. In particular, the number of sub-classes L (see Section 6.1.4 for details) as to be fixed (or possibly chosen via a BIC or ICL criterion). Note that this number can be different for each class. Then click on the "Add" button to run the chosen algorithm and estimate the corresponding parameters. Repeat the same step for class 2 to class K. The table in the learning part summarizes files that are included as training sets. Additionally to segmentation results, a "Learning database" page is displayed in the

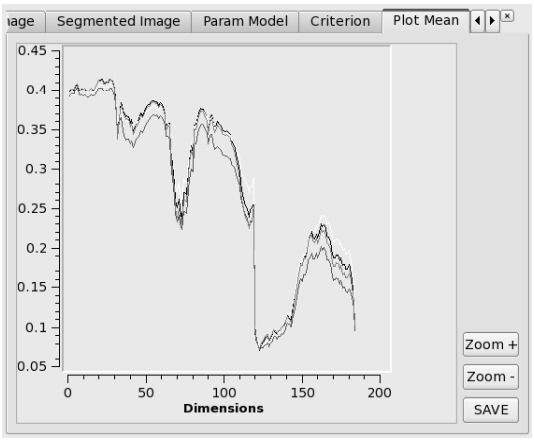


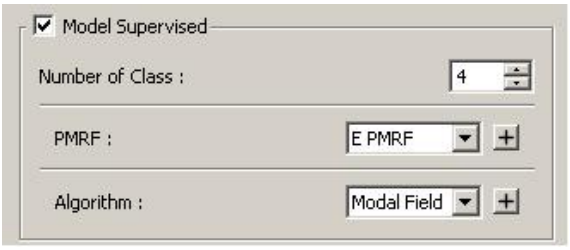
Figure 1: Hyperspectral image of Mars: the first image shows dimension 18 of the data set (left) and the spectrum at coordinates $(300, 126)$ (right). The second image shows mean spectra obtained with the 'Plot Result Mean' option

central part with all used training sets. When all classes are learned, click on the 'End Learning' button to go to the classification step. When doing this the results will be saved in an XML file whose name can be chosen. The interface is now ready for the classification step.

4.3.2 Classification step

At this stage, the software has learnt parameters for each class models using the training sets. It gathers the three main parts that were discussed in the section on unsupervised clustering (Section 4.2). Data from objects to be clustered has to be entered (4.2.1) before setting the model (see Section 4.2.2) and choosing the desired options (Section 4.2.3).

In the supervised segmentation option, models to obtain the final segmentation are more general than those used for unsupervised clustering. Triplet Markov random fields (referred to as "PMRF" in the GUI) is the general model family we consider. See Section 6.2.4 for explanations and a detailed list of available models.

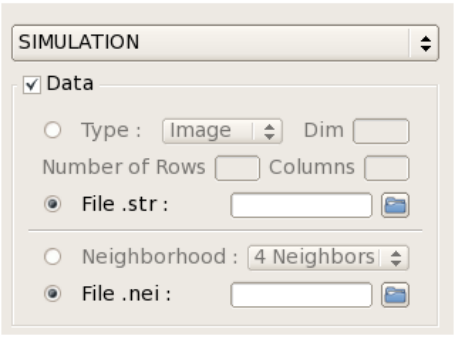


The segmentation can be started by clicking on "RUN" just like in the case of unsupervised clustering. Results are then obtained in a similar way once the computation is over.

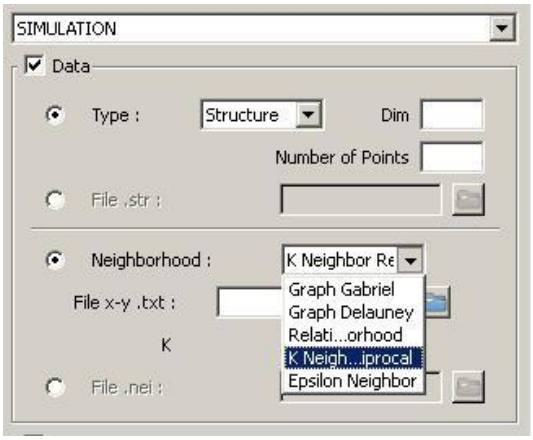
4.4 Simulation

4.4.1 Choosing the data format

The user must first decide on the kind of neighbourhood to simulate data on: either a regular grid ("Image") or a general neighbourhood ("Structure") defined by a graph (see Section 4.2.1). The kind of neighbourhood also needs to be defined. For images on regular 2D grids, available neighborhoods are the standard first and second order neighborhoods.



For non regular graphs, the software proposes a number of standard ways to built a graph from (x,y) coordinates. Once these coordinates are specified via a file, the available graphs are Gabriel, Delaunay, Relative neighborhood, K-neighbor (K has to be specified then) and ϵ (to be specified) graphs. Definitions for these various graphs can be found in [Blanchet, 2007] p. 18-19. The user can also specified is own graph via the specification of the '.nei' file. Then the sample size (numbers of rows and columns for an image) as well as the dimension of the observations have to be set.

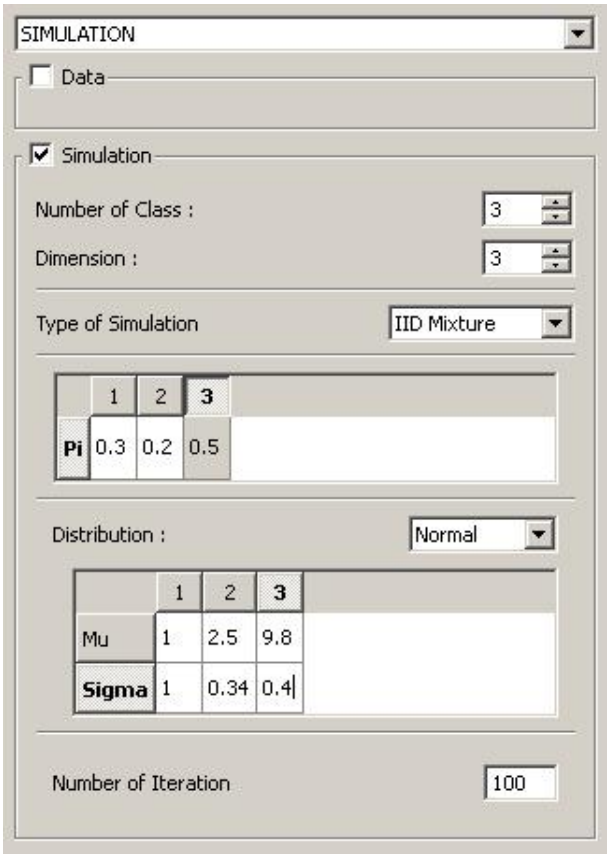


4.4.2 Choosing the simulation type

The number of classes and the dimension of the observations to be simulated have to be specified. SpaCEM^3 can then simulate observations according to one of the following models:

- an independent mixture model, "IID Mixture",
- a pairwise discrete Markov Random Field model (Potts model and various extensions, see Section 6.1.2), "MRF Model",
- an Hidden Markov Random Field which can be seen as a noisy version of a "MRF" model, "HMRF Model",
- a Pairwise Markov Field, "PMRF Model" (see Section 6.2.4) or
- a Triplet Markov Field, which can be seen as a noisy version of a "PMRF" model, "TMRF Model".

For each of these choices, the user has to specify the values of relevant parameters to be used (see Section 6). The image to the right shows a setting to simulate a 3-class Gaussian mixtures. For non "IID" models, the simulation is done via a Gibbs sampler (see [Geman and Geman, 1984]) for the MRF part . In this case the number of iterations (100 by default) to use for the sampler also needs to be specified. For "IID" models, the number of iterations does not need to be specified as it does not play any role in the simulation procedure.



4.4.3 Performing the simulation

The simulation is now parameterized and ready to run ("RUN" button). An XML file has to be specified to store the results which are then displayed on the screen. Two or three pictures will be shown depending on the chosen simulation. When simulating a 'noisy' model, the model without noise is shown as well. Examples of simulations together with the series of actions that provided them are given in Section 7.3.

4.4.4 Saving simulation results

Each of the resulting data sets (*e.g.* simulated images) can be saved in a JPEG file for later reporting or visualization use. The data values themselves are saved, in the current workspace, in files: '.cls' (simulated 'noisy' data) and '.true' (simulated 'data' before adding noise when relevant). This is especially useful when the simulated data are to be used in a subsequent segmentation step as it is often the case when validating models on synthetic data.

5 Command line use of SpaCEM^3 (without the GUI)

5.1 Starting the software

The software can be used without the GUI typing commands at the prompt. The main program can be run with a command of the following type:

```
spacem3 /.../.../essai.xml /.../.../result.xml
```

where '`essai.xml`' is the XML file with data, model and parameter specifications that are necessary to run the computation while '`result.xml`' is the XML file where the results will be stored.

Overview of the XML input file.
The file structure follows the main features of the GUI. Below is an example of a file used in a simple unsupervised clustering case. The data set in file 'damier4.dat' corresponds to a 128×128 grey-level image available at `http://spacem3.gforge.inria.fr`.

```
<!DOCTYPE model (View Source for full doctype...)>
 <model type="UNSUPERVISED SEGMENTATION" name="damier.xml">
  <data type="Spatial Data">
  <path_data>damier4.dat</path_data>
- <define_str type="Image">
  <dimension>1</dimension>
  <nb_row>128</nb_row>
  <nb_col>128</nb_col>
  </define_str>
  <define_nei>4 Neighbors</define_nei>
  </data>
- <seg_model>
  <nb_class>4</nb_class>
- <model type="HMRF"> - <mrf type="B MRF">
  <parameters_mrf choice="Unfix" />
  </mrf>
  </model>
- <distribution type="Normal">
  <parameters_distribution choice="Unfix" />
  </distribution>
- <algorithm type="Simulated Field EM">
  <initialisation>KMeans</initialisation>
  <stopping_criterion>Number of Iterations</stopping_criterion>
  <tolerance>20</tolerance>
  </algorithm>
  </seg_model>
- <options>
  <missing_data method="none" />
  <compute_bic compute="yes" />
  <compute_icl compute="yes" />
  <save_param_model save="yes" />
  <save_param_algo save="yes" />
```

```
<plot_result_mean save="yes" />
</options>
</model>
```

5.2 Retrieving files and results

When running, SpaCEM^3 creates in the current directory output files. These files can be accessed and used directly without the GUI. The list of output files is:

- `test.xml`: XML file containing the model specifications. It stores the tasks to be run by the core software.
- `test_1.xml`: XML file containing the results. It allows the GUI to display them. The '1' is incremented to 2 etc. when running several times the same model as specified in test.xml.
- `test_1.cls`: output segmentation file, with possibly same incrementation as above.
- `test_1.par`: file with final estimated parameter values.
- `test_saveModel.txt`: file that stores values for the estimated model parameters at each iteration of the algorithm (available when the "Save Model Parameters" option has been selected).
- `test_saveAlgo.txt`: file that lists values for the selected criteria (*e.g.* BIC or ICL values) at each iteration (available when the "Save Algo Parameters" option has been selected).
- `test_saveTik.txt`: the values of the final assignment probabilities are stored in this file. Lines correspond to the objects to be classified and columns to the different classes. Each line contains then the assignment probabilities for a given object to each of the classes.

6 Details on SpaCEM^3 functionalities

In this section, we describe into more details the SpaCEM^3 functionalities (models, algorithms, ...), which can all be specified by the user step by step through the GUI (see Sections 2 to 4).

6.1 Models

6.1.1 Independent Mixture Model

The methods implemented in SpaCEM^3 rely on graphical probabilistic models. A widely used probabilistic model for clustering purpose is the *Independent Mixture Model* ("IID" model in the software). Such a model assumes that observations are independent from each other and that their classes to be recovered are also independent. More specifically, let $X_1, \cdots, X_N$ denote the N observations associated to the N objects to be clustered (pixels, genes...). These observations can be uni- or multi-dimensional. Let $(Z_1, \cdots, Z_N)$ denote the N unobserved (or hidden or latent) labels (or classes) associated to these objects. The independent mixture model assumes that:

$$P(Z_1 = z_1, \cdots, Z_N = z_N) = \prod_{i=1}^{N} P(Z_i = z_i) \quad (1)$$

$$\text{and} \quad P(x_1, \cdots, x_n | Z_1 = z_1, \cdots, Z_N = z_N) = \prod_{i=1}^{N} P(x_i | Z_i = z_i) \quad (2)$$

Equation (1) means that classes are statistically independent and equation (2) means that observations are statistically independent conditionally on their classes. It follows from equations (1) and (2) that the X_i's are independent from each other as well. Such an assumption is often too restrictive in many applications such as image segmentation or gene clustering and can lead to poor results (see [Celeux *et al.*, 2003] and [Vignes and Forbes, 2009] for a discussion in the case of image data and gene expression data respectively).

6.1.2 Hidden Markov Random Field

Models that can account for more complex dependence structures are implemented in SpaCEM^3. They are based on the *Hidden Markov Random Field* ("HMRF") model. Interactions are encoded via Markov dependencies between the assignment variables $(Z_1, \cdots, Z_N)$. More specifically, a HMRF model assumes that the field $(Z_1, \cdots, Z_N)$ has a Markovian distribution. In the software, this Markovian distribution takes the form:

$$P(Z_1 = z_1, \cdots, Z_N = z_N) = \frac{1}{W} \exp(\sum_i A_{z_i} + \sum_{i \sim j} Q_{z_i z_j}) \tag{3}$$

where W is a normalizing constant, $i \sim j$ denotes objects that are neighbours (*i.e.* objects directly linked by an edge in the neighbourhood graph), A is a K-dimensional vector of so-called singleton potentials (with K being the total number of classes) and Q is a symmetric $K \times K$ matrix of pairwise interactions. Intuitively, the values in vector A aim at giving different a priori weights to the different classes: the higher A_k, the more likely class k is. Matrix Q accounts for the interaction strength between classes. It can also be seen as a measure of the compatibility between classes: the higher $Q_{kk'}$, the more likely two neighbouring objects are to be classified in these two classes k and k'.

Equation (3) is the most general formulation of discrete MRF distribution with interactions limited to order 2 (pairwise MRF). In SpaCEM^3, 8 different models are derived from equation (3) depending on whether interactions of order 1 are accounted for or not, and depending on the structure of matrix Q:

- The "AkB MRF" model: model with singleton potentials A and isotropic Q matrix. This is sometimes referred to as the *Potts model with external field*,

$$Q = \begin{pmatrix} \beta & & 0 \\ & \ddots & \\ 0 & & \beta \end{pmatrix}$$

- The "B MRF"model: same model as "AkB MRF" without singleton potentials (*i.e.* all components of vector A are set to zero). This corresponds to the standard isotropic *Potts model*.

- The "AkBk MRF" model: model with singleton potentials A and diagonal matrix Q,

$$Q = \begin{pmatrix} \beta_1 & & 0 \\ & \ddots & \\ 0 & & \beta_K \end{pmatrix}$$

- The "Bk MRF" model: same model as "AkBk MRF" without singleton potentials.

- The "AkBGk MRF" model: model with singleton potentials A and full symmetric Q matrix with equal diagonal components,

$$Q = \begin{pmatrix} \beta & & (\gamma_{kk'}) \\ & \ddots & \\ (\gamma_{kk'}) & & \beta \end{pmatrix}$$

  This specification of Q and the following ones can be useful when one wants to penalize neighbours with different classes depending on the classes themselves. In the first two Q models, the penalization is the same (0) whatever the couple of classes outside the diagonal.

- The "BGk MRF" model: same model as "AkBGk MRF" without singleton potentials.

- The "AkBkGk MRF" model: model with singleton potentials A and full Q matrix,

$$Q = \begin{pmatrix} \beta_1 & & (\gamma_{kk'}) \\ & \ddots & \\ (\gamma_{kk'}) & & \beta_k \end{pmatrix}$$

- The "BkGk MRF" model: same model as "AkBkGk MRF" without singleton potentials.

6.1.3 Models for class-dependent distributions

The previous section describes the MRF models available in SpaCEM^3. As regards the specification of a HMRF itself, it is not complete without the specification of the class-dependent distributions, *i.e.* the form of the $P(x_i|Z_i = z_i)$'s in equation (2). Several class-dependent distributions can be used in SpaCEM^3: for class $k \in \{1, \cdots, K\}$,

- Gaussian distribution

$$P(x_i|Z_i = k; \mu_k, \Sigma_k) = \frac{1}{\sqrt{2\pi \det \Sigma_k}} \exp\left(-\frac{(x_i - \mu_k)\Sigma_k^{-1}(x_i - \mu_k)'}{2}\right)$$

  where μ_k is the mean vector and Σ_k is the covariance matrix. Different structures for the covariance matrix Σ_k are proposed:

  - Σ_k is a full matrix ("Normal"),
  - Σ_k is a diagonal matrix ("Diag Normal") or
  - Σ_k is parametrized for high dimensional data: "High Dim Normal" for the general model and "High Dim Normal Const" for a simplified model (see [Bouveyron *et al.*, 2007] and [Blanchet and Forbes, 2008] in a Markovian setting).

- Laplace distribution ("Laplace"):

$$P(x_i|Z_i = k; \mu_k, \sigma_k) = \frac{1}{2\sigma_k} \exp\left(-\frac{|x_i - \mu_k|}{\sigma_k}\right)$$

  where μ_k is a location parameter and $\sigma_k > 0$ is a scaling parameter.

- Poisson distribution ("Poisson"):

$$P(x_i|Z_i = k; \lambda_k) = \exp^{-\lambda_k} \frac{\lambda_k^{x_i}}{x_i!}$$

for x_i being discrete, and the rate λ_k being real.

- Poisson with known risk e_i (acts as a weight) for each site i ("Poisson E"):

$$P(x_i|Z_i = k; \lambda_k) = \exp^{-e_i\lambda_k} \frac{(e_i\lambda_k)^{x_i}}{x_i!}$$

for x_i being discrete, and the rate λ_k being real. This is useful in epidemiology for disease mapping applications. In this case, the relative risks e_i are usually given by the user and have to be specified via a separate file.

6.1.4 Triplet Markov Fields

These models are only available for the 'Simulation' and 'Supervised segmentation' options.

In a number of applications, such as texture modelling and more generally cases where class-dependent distributions cannot assumed to be unimodal, the commonly used independent noise assumption (*i.e.* the conditional independence assumption (2)) is too restrictive and needs to be relaxed. Usually the popularity among users of such an hypothesis comes from the fact that the Markovianity of the conditional distribution of the classes given the observations (the posterior distribution) is then guaranteed. And this is the key to parameter estimation and class assignment based on standard Bayesian techniques. However, it appears that the two standard HMRF hypothesis, namely the Markov property of the hidden classes (eq. (3)) and the observations conditional independence (eq. (2)), are sufficient but not necessary for the above Markovianity to hold. This simple observation led to a new family of Markov models, the so-called Pairwise Markov random fields, introduced in [Pieczynski and Tebbache, 2000] and then extended to Triplet Markov fields in [Benboudjema and Pieczynski, 2005]. Such models have the advantage to allow richer class-dependent distributions (noise models) for equivalent algorithmic costs. The Triplet Markov models available in SpaCEM^3 are those described in [Blanchet and Forbes, 2008]. They differ from those in [Benboudjema and Pieczynski, 2005, Benboudjema and Pieczynski, 2007]. They have been initially designed for the supervised classification of observations generated from non standard class-dependent distributions, namely non-unimodal or non independent distributions. More details can be found in [Blanchet and Forbes, 2008].

Using the same notation as in Section 4.3, we focus on data that exhibit some complexity due to the general nature of the noise ($p(\mathbf{x}|\mathbf{z})$) that does not necessarily satisfy usual standard assumptions such as being Gaussian and non correlated. When starting from the standard HMRF models, referred to as HMF-IN (for Hidden Markov Field with Independent Noise) in what follows, to cluster data, a natural idea to extend the modelling capabilities of this approach is to decompose each class, given by the z_i's, into subclasses allowing this way more general class dependent distributions. Let us first assume that each of the K classes is decomposed into L sub-classes so that we can introduce additional variables $\{Y_1, \ldots, Y_n\}$ indicating the sub-classes and then consider class and sub-class dependent distributions $P(.|y_i, z_i)$ that depend on some parameters $\theta_{y_i z_i}$. To account for dependencies between objects both in the learning and test step, we consider a Triplet Markov model that is a triplet $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ which has a Markovian joint distribution. The triplet models in

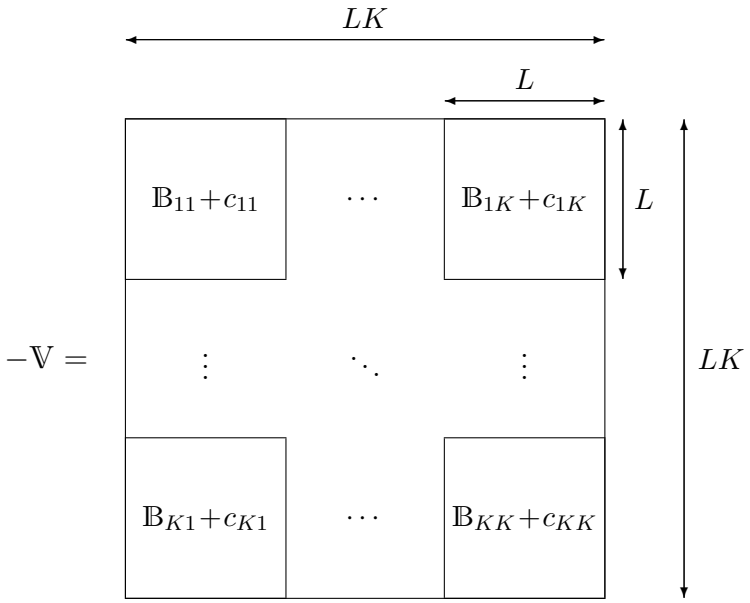


Figure 2: $-\mathbb{V}$ matrix

SpaCEM^3 have the following form:

$$P_G(\mathbf{x},\mathbf{y},\mathbf{z}) \propto \exp(-\sum_{i\sim j} V_{ij}(y_i,z_i,y_j,z_j) + \sum_{i\in\mathcal{I}} \log f(x_i|\theta_{y_i z_i})) \tag{4}$$

where the V_{ij} are pair potentials which are assumed not to depend on i and j so that we can write:

$$V_{ij}(y_i,z_i,y_j,z_j) = -B_{z_i z_j}(y_i,y_j) - C(z_i,z_j)$$

where the $B_{kk'}$ are K^2 functions that map $\mathcal{L}\times\mathcal{L}\to\mathbb{R}$ and C is a function that maps $\mathcal{K}\times\mathcal{K}\to\mathbb{R}$. Using the vectorial notation $z_i = k \Leftrightarrow \mathbf{z}_i = \mathbf{e}_k$ and $y_i = l \Leftrightarrow \mathbf{y}_i = e'_l$ we can further write:

$$V_{ij}(\mathbf{y}_i,\mathbf{z}_i,\mathbf{y}_j,\mathbf{z}_j) = -\mathbf{y}'_i \mathbb{B}_{z_i z_j}\mathbf{y}_j - \mathbf{z}'_i\mathbb{C}\mathbf{z}_j \tag{5}$$

where the $\mathbb{B}_{kk'}$ are K^2 $L\times L$ matrices and $\mathbb{C}$ is a $K\times K$ matrix. This way of writing (5) is always possible. V_{ij} is seen as a $LK\times LK$ matrix $\mathbb{V}$, which is decomposed into $L\times L$ bloc matrices of size $K\times K$. Denoting the $\mathbb{C}$ coefficients by $(c_{kk'})_{k,k'\in\mathcal{K}}$, the form of matrix $\mathbb{V}$ is the one given in Figure 2.

As interactions are symmetric, $\mathbb{V}$ is symmetric and thus the $\mathbb{B}_{kk}$ matrices are too. In SpaCEM^3, it is assumed in addition that $\mathbb{C}$ is symmetric so that all $\mathbb{B}_{kk'}$ matrices are symmetric.

The reason for such a parameterization is made clearer below, in the classification step. The (l,l') term in matrix $\mathbb{B}_{kk'}$ can be interpreted as measuring the compatibility between sub-class l of class k and sub-class l' of class k'. The larger this term, the more likely are neighboring sites to be in such sub-classes. Similarly the (k,k') term in $\mathbb{C}$ has also to do with the compatibility between classes k and k'. The higher this term, the more likely are classes k and k' to be neighbors. When $\mathbb{C} = c\mathbb{I}_K$, the only parameter is $c\in\mathbb{R}^+$ and the term $\sum_{i\sim j}\mathbf{z}'_i\mathbb{C}\mathbf{z}_j = c\sum_{i\sim j}\mathbb{1}_{z_i=z_j}$ acts then as a regularizing term favoring homogeneous regions (connected groups of objects in the same class). This is the model denoted by **E PMRF** in the software interface (see Section 6.2.4).

Note that if we denote by $\mathbf{U}$ the couple $(\mathbf{Y}, \mathbf{Z})$, it follows from equation (4) that $\mathbf{U}$ is a Markov field and that $(\mathbf{X}, \mathbf{U})$ is an HMF-IN. This later field is used in the classification step (Section 6.2.2). Also from (4), $P(\mathbf{y}|\mathbf{z})$ is Markovian too:

$$P(\mathbf{y}|\mathbf{z}) = \frac{1}{W(\mathbf{z})} \exp(\sum_{i\sim j} \mathbf{y}_i' \mathbb{B}_{z_i z_j} \mathbf{y}_j) \tag{6}$$

where the normalizing constant $W(\mathbf{z})$ depends on $\mathbf{z}$. Using (4), it comes

$$P_G(\mathbf{x}, \mathbf{y}|\mathbf{z}) = \frac{1}{W(\mathbf{z})} \exp(\sum_{i\sim j} \mathbf{y}_i' \mathbb{B}_{z_i z_j} \mathbf{y}_j + \log f(x_i|\theta_{y_i z_i})), \tag{7}$$

so that conditionnaly to $\mathbf{Z} = \mathbf{z}$, $(\mathbf{X}, \mathbf{Y})$ is an HMF-IN for which we can apply standard estimation techniques. This later field is used in the learning step.

Illustration. A simple example of such a triplet model is given by:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{z}) \propto \exp(b \sum_{i\sim j} \mathbb{1}_{y_i=y_j} \mathbb{1}_{z_i=z_j} + c \sum_{i\sim j} \mathbb{1}_{z_i=z_j} + \sum_{i\in\mathcal{I}} \log f(x_i|\theta_{y_i z_i})) \tag{8}$$

where b and c are real parameters, $\theta_{lk} = (\mu_{lk}, \Sigma_{lk})$ are the parameters of Gaussian distributions $f(.|\theta_{lk})$, for $l \in \mathcal{L}$ and $k \in \mathcal{K}$. The couple $(\mathbf{Y}, \mathbf{Z})$ is then Markovian:

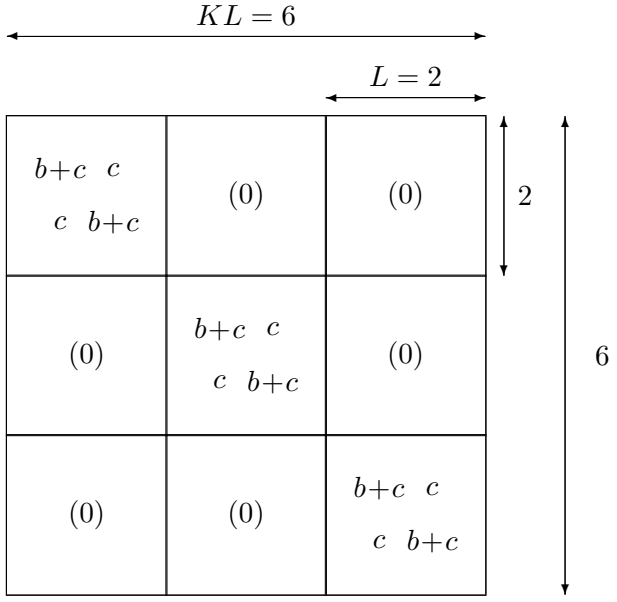
$$P(\mathbf{y}, \mathbf{z}) \propto \exp(b \sum_{i\sim j} \mathbb{1}_{y_i=y_j} \mathbb{1}_{z_i=z_j} + c \sum_{i\sim j} \mathbb{1}_{z_i=z_j}) \, . \tag{9}$$

When comparing to equation (5) this is equivalent to:

- for all $k \in \mathcal{K}$, $\mathbb{B}_{kk} = b\mathbb{I}_L$ and all $k' \neq k$, $\mathbb{B}_{kk'} = 0_L$ (The null matrix)
- matrix $\mathbb{C}$ is diagonal, with all non zero components equal to c.

For $K = 3$ and $L = 2$, matrix $\mathbb{V}$ in Figure 2 becomes then

$-\mathbb{V} =$



Different particular cases can be specified:

- when $L = 1$, the model in (9) is a K-class Potts model with interaction parameter $b + c$.
- when $K = 1$, the model is a L-class Potts model with interaction parameter b.
- when $c = 0$, the model is a LK-class Potts model with interaction parameter b.

In Section 7, Figure 7 shows simulations from a Triplet Markov model defined by (8) for $K = L = 2$ and different b and c values. Each of the $LK = 4$ possible couple (y_i, z_i) is associated to a grey-level for visualization purpose.

6.2 Algorithms

6.2.1 Estimation algorithms

For IID models, SpaCEM^3 proposes 4 estimation algorithms. Note that among them, the NEM and NCEM algorithms can actually account for neighborhood structures and do not really correspond to IID models in that sense:

- the EM algorithm (Expectation-Maximization, see [Dempster *et al.*, 1977])
- the CEM algorithm (Classification-Expectation-Maximization, see [Celeux and Govaert, 1992])
- the NEM algorithm (Neighbourhood-Expectation-Maximization, see [Ambroise *et al.*, 1997])
- the NCEM algorithm (Neighbourhood-Classification-Expectation-Maximization)

The EM algorithm is an iterative method. At each iteration, it alternates between performing (i) an Expectation (E) step, which computes the expectation of the log likelihood with respect to the current estimate of the distribution (parameters known from previous step) and (ii) a Maximization (M) step, which computes the parameters maximizing the expected log likelihood found in the E step. The CEM algorithm adds a Classification (C) step in between the E and the M steps. The NEM algorithm allows to add spatial information in the estimation. The NCEM algorithm includes both features.

For HMRF models, SpaCEM^3 proposes the following estimation algorithms:

- ICM algorithm (Iterated Conditional Modes, see [Besag, 1986])
- Mean field-like EM algorithms (see [Celeux *et al.*, 2003]) that include three variants:
  - Mean Field EM
  - Modal Field EM
  - Simulated Field EM

6.2.2 Supervised clustering scheme for Triplet Markov models

More than an algorithm, we describe a general scheme to deal with complex data (non independent noise models, non-unimodal class dependent distributions) using Triplet Markov field (TMF) models. For such models, starting from a description of a supervised clustering issue in terms of K *complex* classes corresponding to some non standard noise model, the learning step can be decomposed into K simpler issues, each involving a standard HMRF model with L classes (referred to as

an HMF-IN for hidden Markov field with independent noise). The following classification step can then be reduced to an inference problem for a standard HMF-IN model with KL classes. It follows that the computational complexity of the TMF models may vary depending on the algorithm chosen for inference but is equivalent to that of usual HMF-IN models.

6.2.3 Learning step

In this stage, for a number of individuals $i \in \mathcal{I}^1$, we both observe x_i and its corresponding class z_i. Using the triplet model defined in Section 6.1.4, it remains that the y_i are missing. We can apply one of the algorithms available for HMRFs (Section 6.2.1) to the HMF-IN $(\mathbf{X}, \mathbf{Y}|\mathbf{Z} = \mathbf{z})$ to provide estimates of the model parameters which are the $\mathbb{B}_{kk'}$, $k, k' \in \mathcal{K}$ matrices and the θ_{lk}, for $l \in \mathcal{L}$ and $k \in \mathcal{K}$. Estimating the later parameters is especially important to solve identifiability issues when dealing with our Triplet Markov fields in the classification step. To estimate the θ_{lk}'s it is necessary that all K classes are sufficiently represented in the learning data. In practice the learning data are often divided in a number of separate data sets so that the learning procedure actually consists of a number of separate runs of the estimation algorithm. As regards the $\mathbb{B}_{kk'}$'s estimated in the learning stage, we do not necessarily need to keep them for the classification step. However, for complex data, it may be that learning also the $\mathbb{B}_{kk'}$'s or at least part of them is a better choice in terms of modelling capabilities. Typically, if the underlying neighborhood structure is such that there exists no neighbors in classes k and k', then $\mathbb{B}_{kk'}$ cannot be estimated since terms involving $\mathbb{B}_{kk'}$ will not appear in the model formulas. More generally if the number of pairs in classes k and k' is too small, the estimation of $\mathbb{B}_{kk'}$ is likely not to be good and in this case it would be better to ignore it. According to equation (6), learning class k reduces then to estimate a L-class Potts model with compatibility matrix $\mathbb{B}_{kk}$.

6.2.4 Classification step

At this stage, $\mathbf{x}^2$ for $i \in \mathcal{I}^2$ are not labelled so that both fields $\mathbf{Y}$ and $\mathbf{Z}$ are missing. Defining $\mathbf{U} = (\mathbf{Y}, \mathbf{Z})$, it comes that $(\mathbf{X}, \mathbf{U})$ is an HMF-IN for which the usual Bayesian techniques can be applied. The parameters are the K^2 $L \times L$ dimensional matrices $\{\mathbb{B}_{kk'},\, k, k' \in \{1, \ldots, K\}\}$ and the LK parameters $\{\theta_{lk}, l = 1 \ldots L, k = 1 \ldots K\}$ with in addition the $K \times K$ dimensional matrix $\mathbb{C}$, *ie.* parameters $\{\mathbb{C}_{kk'},\, k, k' \in \{1, \ldots, K\}\}$. At this stage, the θ_{lk}'s have to be considered as fixed to the values computed in the learning stage. For the $\mathbb{B}_{kk'}$'s, different strategies arise depending on the available learning data and the goal in mind, in particular the type of interactions we want to account for. These matrices can be either entirely fixed, only partially fixed or entirely re-estimated. In practice, we propose to use specified or learned values for all $\mathbb{B}_{kk'}$'s. In most cases then, regarding parameters, the classification step consists of estimating $\mathbb{C}$. The classification step differs then in that $\mathbb{C}$ plays an important role in modelling interaction. One possibility is to specify $\mathbb{C}$ to be of the Potts form *ie.* to consider diagonal $\mathbb{C}$. More complex knowledge on the classes could be incorporated through other definitions of $\mathbb{C}$ but this simple case appears satisfying in a number of applications. Although $\mathbf{Y}$ is not Markovian, this acts as a regularizing term favoring homogeneous regions of the same class. This is an important feature of our classification step.

In the current version of the software the available models correspond to the estimation of matrix $\mathbb{C}$ only. It follows that 6 different triplet models are currently available depending on whether interactions of order 1 are accounted for or not, and depending on the structure of the matrix $\mathbb{C}$:

- Model "DkE PMRF": model with singletons D and a constant diagonal matrix $\mathbb{C} = cI_K$

$$\mathbb{C} = \begin{pmatrix} c & & 0 \\ & \ddots & \\ 0 & & c \end{pmatrix}$$

- Model "E PMRF": same model as "DkE PMRF" without singletons (*i.e.* all components of vector D are set to zero).

- Model "DkEk PMRF": model with singletons D and diagonal matrix $\mathbb{C}$

$$\mathbb{C} = \begin{pmatrix} c_1 & & 0 \\ & \ddots & \\ 0 & & c_K \end{pmatrix}$$

- Model "Ek PMRF": same model as "DkEk PMRF" without singletons.

- Model "DkEkNk PMRF": model with singletons D and full $\mathbb{C}$ matrix

$$\mathbb{C} = \begin{pmatrix} c_1 & & (\nabla_{kk'}) \\ & \ddots & \\ (\nabla_{kk'}) & & c_K \end{pmatrix}$$

- Model "EkNk PMRF": same model as "DkEkNk PMRF" without singletons.

An illustration of the flexibility of such models is given in Section 7.2.

6.3 Model selection

For IID models, two criteria can be used for model selection:

- the Bayesian Information Criterion ("BIC", see [Schwarz, 1978]) and
- the Integrated Completed Likelihood ("ICL", see [Biernacki *et al.*, 2000]) criterion.

For HMRF model, these two criteria cannot be computed exactly and approximations are needed. The following mean field-like approximations are used in SpaCEM^3 (see [Forbes and Peyrard, 2003]):

- "BICp": Bayesian Information Criterion with approximation of the Markovian distribution,
- "BICw": Bayesian Information Criterion with approximation of the Markovian normalizing constant,
- "ICLp": Integrated Completed Likelihood criterion with approximation of the Markovian distribution and
- "ICLw": Integrated Completed Likelihood criterion with approximation of the Markovian normalizing constant.

These selection criteria can be selected in the "Options" menu (see Section 4.2.3).

6.4 Dealing with missing data

With SpaCEM^3, the classification can still be performed in a reliable manner when some observations are missing (see [Blanchet and Vignes, 2009]). In the data file `'.dat'` (see Section 4.2.1), missing values have to be spotted with `NaN` or `NA`. For example, if the second observation is missing for site 1, and the third observation is missing for site 3, the data file will look like:

```
x11 NaN x13 ... x1d
x21 x22 x23 ... x1d
n31 x32 NaN ... x3d
```

To cluster such data, the following methods can be used (to be selected in the "Options" menu, see Section 4.2.3):

- *off-line imputation.* It corresponds to a preprocessing step previous to the estimation of the model by one of the algorithms mentioned above. A missing value x_{id} can be imputed by either the:
  - mean of the d^{th} component of the k most similar observations ("KNN Imputation", standing for k-nearest neighbours),
  - mean of the d^{th} component of all observations ("Mean Imputation" option),
  - mean of the observed values for site i ("Mean Row imputation" option),
  - median of the d^{th} component of all observations ("Median Imputation" option),
  - median of the observed values for site i ("Median Row imputation" option),
  - predicted value obtained by linear regression on observed data ("Reg Imputation" option),
  - value 0 ("Zero Imputation" option) or
  - value x_{id-1} ("Previous Imputation" option also called *last observation carried forward* in the literature)
- *on-line imputation.* In this case, a probabilistic estimation of the missing observations is performed at each iteration of the algorithm. At iteration (q), when updating parameter $\theta_k^{(q)}$ of the distribution of class k $P(Z_i = k; \theta_k)$ in the E-step, a missing value x_{id} can be imputed by either the:
  - unconditional mean $\mu_k^{(q-1)}$ of class k, *i.e.* by the mean of distribution $P(X_{id}; \theta_k^{(q-1)})$ ("UMI Imputation") or by the
  - conditional mean $\eta_{id}^{(q-1)}$ of class k, *i.e.* by the mean of distribution $P(X_{id}|x_{io_i}; \theta_k^{(q-1)})$ where x_{io_i} denotes all observed values for site i ("CMI Imputation").

Both off-line and on-line imputation methods introduce a bias in the estimation. To tackle this issue, it is possible to classify data without applying any imputation ("No Imputation") using the EM algorithm principle. Indeed the standard EM algorithm can be adapted easily by including missing observations as additional missing data [Little and Rubin, 2002]. This extension has been adapted to HMRF settings in [Blanchet, 2007]. Both cases are implemented in SpaCEM^3. Note that, if one is eventually interested in recovering missing observations, a final imputation is still possible with this procedure and is far more reliable than other approaches. Details and applications to simulated and molecular biological datasets can be found in [Blanchet and Vignes, 2009].

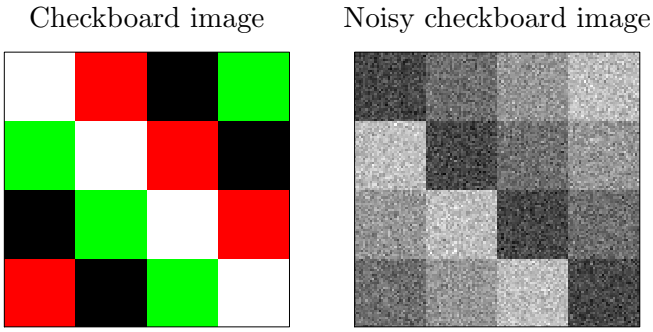


Figure 3: The checkboard image (a) and its noisy version (b) that corresponds to the data in file 'damier2_s01_d1.dat'

7 Examples

Example data sets are available with the software at `http://spacem3.gforge.inria.fr` in Section 'Example data sets'.

7.1 Unsupervised segmentation example

Data in '`damier2_s025_d1.dat`' correspond to a 4 class 128×128 checkboard image (Figure 3 (a)) degraded with an independent Gaussian noise with variance $\sigma^2 = 0.25$ (Figure 3 (b)). A segmentation result close to that in Figure 4 can be recovered as follows:

- "Data" group: '`damier2_s025_d1.dat`', '`damier2_s025_d1.str`' for the structure (image with 128 rows and 128 columns), '`damier2_s025_d1.nei`' for the neighborhood (8-nearest neighbors), or equivalently this information can be set manually,
- "Model" group: 4-class "HMRF" model with Potts Markovian distribution ("B MRF") and 1D Gaussian class-dependent distribution ("Normal")
- "Algorithm" group: "Simulated Field" initialized by "K-means" and stopped after 100 iterations.

The estimated parameters saved by SpaCEM^3 are in '`damier2_s025_d1_1.par`'. The first line correspond to the Markovian parameters (here only one value β), then lines 2 and 3 are respectively the mean μ_0 and variance σ_0^2 for the Gaussian distribution of the first class (labelled 0), lines 4 and 5 are the mean μ_1 and variance σ_1^2 for the second class, ... For instance, in our run, we got the following results (note the order can vary and the estimation be slightly different):

	β	μ_0	σ_0^2	μ_1	σ_1^2	μ_2	σ_2^2	μ_3	σ_3^2
True	-	0	0.25	1	0.25	2	0.25	3	0.25
Estimated	2.09	0.01	0.25	0.99	0.26	2.00	0.26	3.01	0.25

The final clustering result is in '`damier2_s025_d1_1.cls`' (labels range in $\{0, 1, 2, 3\}$ for the 4 classes). Then, all the results can directly be loaded in SpaCEM^3 using the "Load Results" option in the "File" menu and selecting '`damier2_s025_d1_1.xml`'.

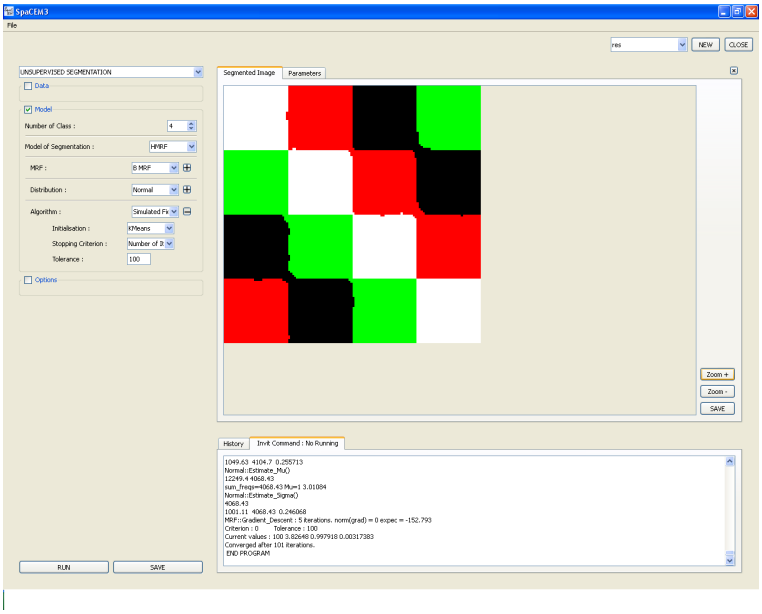


Figure 4: Example of an unsupervised segmentation result for data set 'damier2_s02_d1.dat'

7.2 Supervised segmentation example

Figure 6 shows the superior performance of Triplet Markov models over standard HMRFs. It shows a synthetic 2-class image and two noisy versions using different noise models (non unimodal and correlated). In [Blanchet and Forbes, 2008], an application to texture recognition is also reported.

For illustration we focus on image Fig. 6 (c). Steps are similar for images in Fig. 6 (a,b,d). The image in Fig. 6 (c) is a 128×128 grey-level image made of the two classes whose locations are shown in the "True Segmentation" image. The observations from class 1 are generated from a Gamma(1,2) distribution and the observations from class 2 are obtained by adding 1 to realizations of an Exponential distribution with parameter 1. The file containing image (c) is '`cible_m0S1.dat`'. However, in a supervised segmentation case, the first learning step requires files representing each class, here class 1 and class 2. The latter correspond respectively to files '`app_m0S1.dat`' and '`app_m1S1.dat`' which are also both 128×128 images but note that this needs not to be the case in practice.

Learning models for classes 1 and 2.

For simplicity, we will assume that each class can be modelled using a $L = 4$-class HMRF model. This value is the one selected by BIC criteria for classes 1 and 2 (see Section 6.3 and illustration Figure 5). For class 1, SpaCEM^3 can be set as follows:

After choosing the 'Supervised segmentation' option, set:

- "Data" group: '`app_m0S1.dat`' (1 dimensional image with 128 rows and 128 columns), with a 2nd order neighborhood (8-nearest neighbors),
- "Model" group: 4-class "HMRF" model with Potts Markovian distribution ("B MRF") and 1D Gaussian class-dependent distribution ("Normal")
- "Algorithm" group: "Simulated Field" initialized by "K-means" and stopped after 100 iterations.

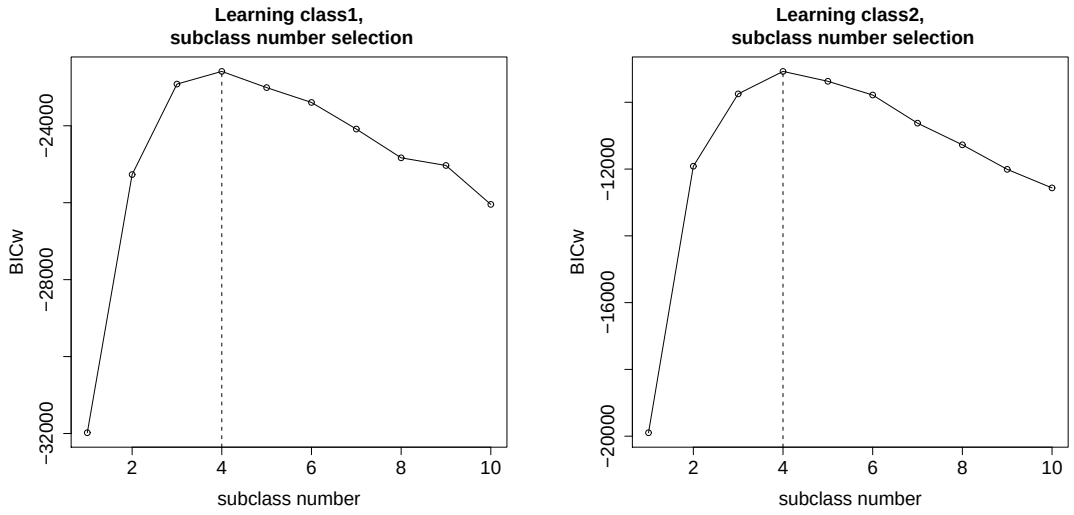


Figure 5: Subclass number selection for classes 1 and 2. $L = 4$ is selected for both classes.

Then click on the 'Add' button to run the estimation (learning) of this class.

For class 2, repeat the same steps with file '`app_m1S1.dat`'. Note that the chosen number of sub-classes L, the model and algorithm need not to be the same as for class 1. Click on 'Add' again to perform the estimation procedure.

When all classes are learned, click on the 'End learning' button. This saves the results of the learning stage in an XML file.

Segmentation step. A possible setting is

- "Data" group: '`cible_m0S1.dat`' (1 dimensional image with 128 rows and 128 columns), with a 2nd order neighborhood (8-nearest neighbors),
- "Model" group: a 'PMRF' 2-class Triplet model with "EkNk PMRF" to specify the interaction model chosen (only Gaussian class-dependent distributions have been implemented at this stage).
- "Algorithm" group: "Simulated Field" initialized by "K-means" and stopped after 100 iterations.

The final clustering result is in '`cible_m0S1.cls`' (labels range in $\{0, 1\}$ for the 2 classes).

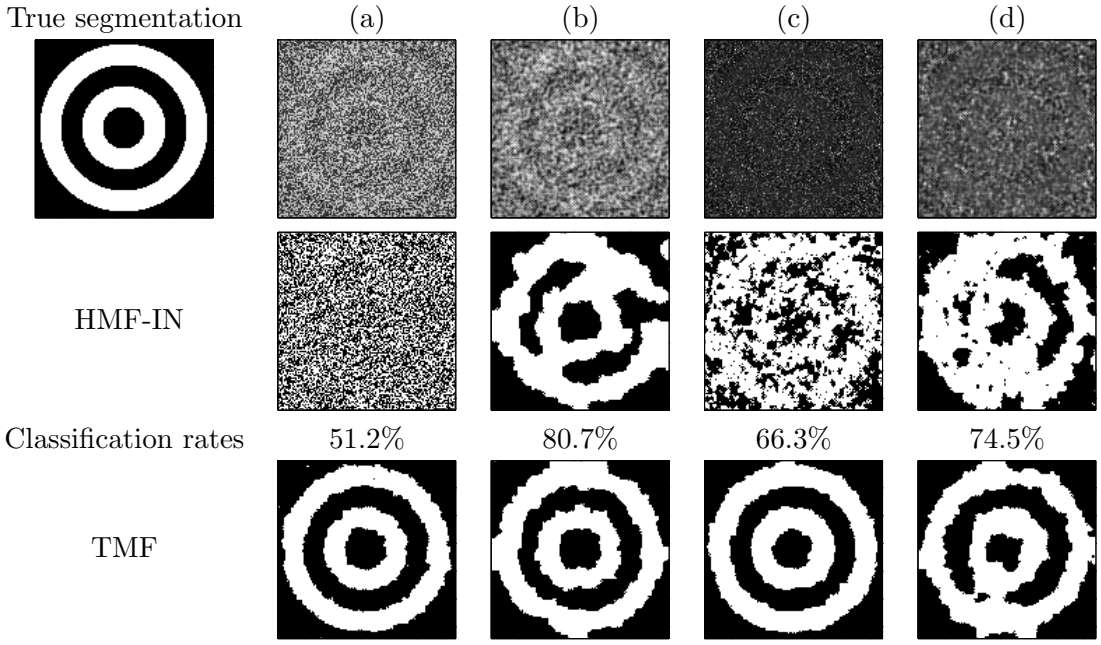
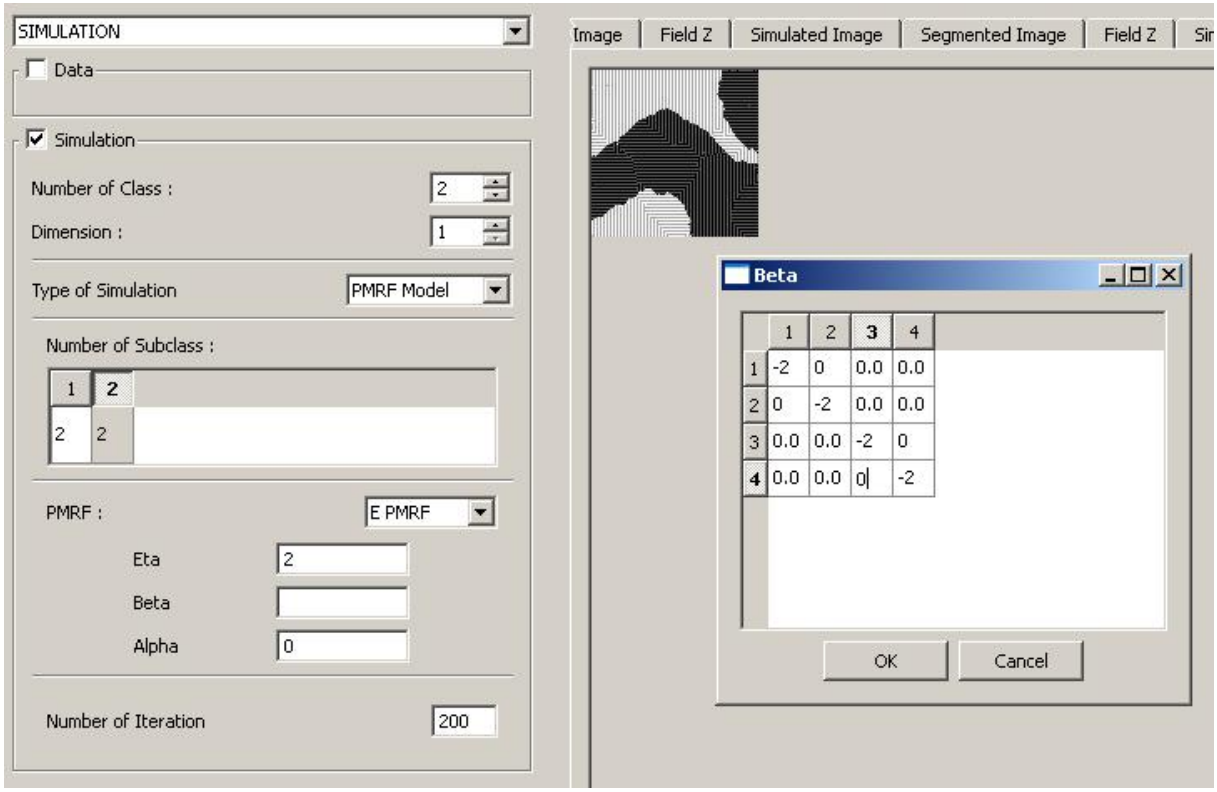


Figure 6: Synthetic image segmentations using an HMF-IN model (second row) and a TMF model (third row): the true 2-class segmentation is the image in the upper left corner and four different noise models are considered. In (a) class distributions are mixtures of two Gaussians, In (c) observations from class 1 are generated from a Gamma(1,2) distribution and observations from class 2 are obtained by adding 1 to realizations of an Exponential distribution with parameter 1. In (b) and (d) the noisy images are obtained by replacing each pixel value respectively in (a) and (c) by its average with its four nearest neighbors. Classification rates are given below each segmentation results. In the TMF model case, the selected L values using the BIC_{MF} criterion is given in the last row.

7.3 Simulation example

Figure 7 shows examples of simulated data from the Triplet model defined in equation (8) for $K = L = 2$ and different b and c values. Each of the 4 possible values of (y_i, z_i) has been associated to a grey-level for visualization.
These simulations have been obtained using the following specification in SpaCEM^3. In the 'data' menu, set the dimension to 1, the image size to 128×128, and the neighborhood to 8 neighbors. In the 'simulation' menu, choose 2 classes and the 'E PMRF' model. Choose $L = 2$ for the number of sub-classes of each of the two classes. Three parameters (2 matrices and 1 vector) have then to be fixed, 'Eta', 'Beta' and 'Alpha'. Parameter 'Eta' is the $K \times K$ $\mathbb{C}$ matrix which for the 'E PMRF' option is diagonal with equal diagonal components, 'Beta' corresponds to the K^2 matrices $\mathbb{B}_{kk'}$ of size $L \times L$. In SpaCEM^3, $\mathbb{B}_{kk'} = 0$ when $k \neq k'$. It follows that only two matrices $\mathbb{B}_{11}$ and $\mathbb{B}_{22}$ need to be fixed here. For the triplet model in eq. (8), these matrices are diagonal with diagonal components equal to b. Then 'Alpha' refers to an external field parameter which consists of a KL-dimensional vector. All components are set to 0 in this case. The picture below illustrates the actions to get images similar to images (a) and (b) of the first row in Figure 7.



References

[Ambroise *et al.*, 1997] Ambroise, C., Dang, M. and Govaert, G. (1997), Clustering of spatial data by the EM algorithm. *Geostatistics for Environmental Applications*, 493–504, ed: Kluwer Academic Publisher.

[Benboudjema and Pieczynski, 2005] Benboudjema, D. and Pieczynski, W. (2005), Unsupervised image segmentation using Triplet Markov fields. *Comput. Vision Image Underst.* **99**, 476–498.

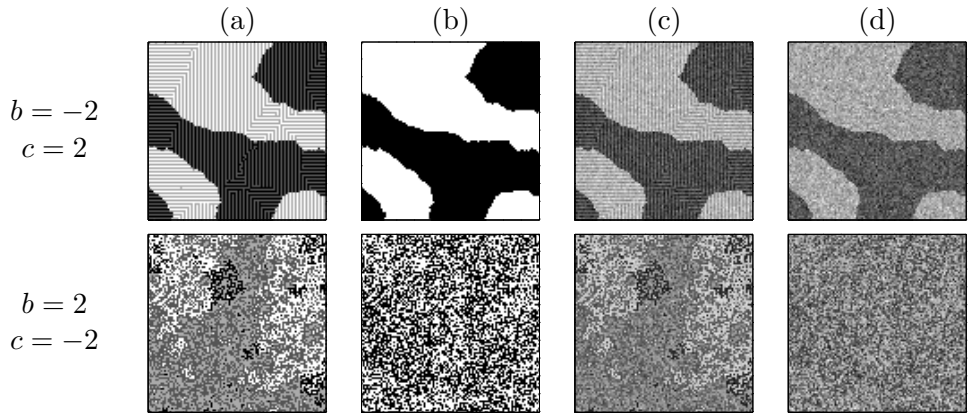


Figure 7: Simulations of 2 parameter (b and c) TMF defined by (8) when $L = 2$ and $K = 2$ with respectively $b = -2$, $c = 2$ (first row) and $b = 2$, $c = -2$ (second row): (a) Realizations of $(\mathbf{Y}, \mathbf{Z})$, (b) Realizations of $\mathbf{Z}$, (c) Realizations of $\mathbf{X}$, (d) Realizations of a HMF-IN built by adding to images in (b) some Gaussian noise with 0 mean and standard deviation equal to 0.3. Note that in the images, each of the 4 possible values of (y_i, z_i) has been associated to a grey-level for visualization.

[Benboudjema and Pieczynski, 2007] Benboudjema, D. and Pieczynski, W. (2007), Unsupervised statistical segmentation of non stationary images using triplet Markov Fields. *IEEE Trans. PAMI* **29**(8), 367–1378.

[Besag, 1986] Besag, J. (1986), On the statistical analysis of dirty pictures. *J. Roy. Stat. Soc. B* **48**, 259–302.

[Biernacki *et al.*, 2000] Biernacki, C., Celeux, G. and Govaert, G. (2000), Assessing a mixture model for clustering with the integrated complete likelihood. *IEEE PAMI* **22**, 719–725.

[Blanchet, 2007] Blanchet, J. (2007), *Modèles markoviens et extensions pour la classification de données complexes.* PhD thesis, Université Grenoble 1.

[Blanchet and Forbes, 2008] Blanchet, J. and Forbes, F. (2008), Triplet Markov fields for the supervised classification of complex structure data. *IEEE PAMI* **30**, 1055–67.

[Blanchet and Vignes, 2009] Blanchet, J. and Vignes, M. (2009), A model-based approach to gene clustering with missing observations reconstruction in a Markov Random Field framework. *J. Comput. Biol.* **16**, 475–86.

[Bouveyron *et al.*, 2007] Bouveyron, C., Girard, S. and Schmid, C. (2007), High dimensional data clustering. *Comput. Statist. Data Analysis* **52**, 502–519.

[Celeux and Govaert, 1992] Celeux, G. and Govaert, G. (1992), A classification EM algorithm for clustering and two stochastic versions. *Comput. Stat. Data Ana.* **14**, 315–332.

[Celeux *et al.*, 2003] Celeux, G., Forbes, F. and Peyrard, N. (2003), EM procedures using mean field-like approximations for Markov model-based image segmentation. *Pat. Rec.* **36**, 131–144.

[Dempster *et al.*, 1977] Dempster, A., Laird, N.M. and Rubin, D.B. (1977), Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B* **39**(1), 1–38.

[Forbes and Peyrard, 2003] Forbes, F. and Peyrard, N. (2003), Hidden Markov random field model selection criteria based on mean field-like approximations. *IEEE PAMI* **25**, 1089–1101.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984), Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE PAMI* **6**(6), 721–741.

[Little and Rubin, 2002] Little, R.J. and Rubin, D.B. (2002), *Statistical analysis with missing data.* New-York: Wiley, second edition.

[Pieczynski and Tebbache, 2000] Pieczynski, W. and Tebbache, A (2000), Pairwise Markov Random Fields and segmentation of textured images. *Machine Graph. Vision*, **9**, 705–718.

[Schwarz, 1978] Schwarz, G. (1978), Estimating the dimension of a model. *Ann. Stat.* **6**, 461–464.

[Vignes and Forbes, 2009] Vignes, M. and Forbes, F. (2009), Gene clustering via integrated Markov models combining individual and pairwise features. *IEEE/ACM TCBB* **6**, 260–270.